Indonesian Journal of Electrical Engineering and Computer Science
Vol. 41, No. 1, January 2026, pp. 73~82
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v41.i1.pp73-82

An energy-efficient hardware module for edge detection using XNOR-Popcount in resource-constrained devices

Van-Khoa Pham, Lai Le
Ho Chi Minh City University of Technology and Education, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Jul 15, 2025
Revised Dec 12, 2025
Accepted Dec 22, 2025

Keywords:

Approximate computing
Edge detection
Low-power hardware design
Multiply-accumulator
XNOR-Popcount

ABSTRACT

Edge detection is a fundamental building block in many embedded vision tasks, including drone navigation, IoT cameras, and wearable devices. However, traditional edge detectors based on multiply–accumulate (MAC) operations are poorly suited to the tight power and area budgets of such resource-constrained hardware. This work introduces a fully synthesizable Prewitt edge detector that replaces MAC operations with 1-bit XNOR–Popcount logic. Incoming 8-bit pixels and ±1 kernel coefficients are binarized, processed by parallel XNOR gates, and tallied by a lightweight Popcount adder tree, eliminating all multipliers and DSP slices. Prototyped on a Xilinx Zynq-7020 FPGA, the proposed design reduces lookup-table usage by 55% and flip-flop count by 26%, cuts dynamic power by about 60%, and supports clock frequencies up to five times higher than a MAC-based core. Frame-level evaluations on the MNIST and ORL datasets show near-lossless edge fidelity, with per-image dissimilarity scores below 0.08 and throughput gains approaching four times. These results demonstrate that hardware-aware binary approximations can enable real-time, energy-efficient edge detection for embedded AI systems without sacrificing functional accuracy.

Corresponding Author:

Van-Khoa Pham
Ho Chi Minh City University of Technology and Education
Ho Chi Minh City, Vietnam
Email: sitizaiton@umk.edu.my

1. INTRODUCTION

Edge computing enables real-time image processing by moving computation closer to data sources, reducing latency and improving privacy. Convolutional operations, such as edge detection and feature extraction, are core to many vision tasks but are difficult to implement efficiently on resource-constrained edge devices due to their high computational cost, limited memory bandwidth, high power consumption, and latency. Overcoming these issues is essential for real-time AI on IoT, embedded, and mobile platforms. The main bottleneck is the large number of multiply-accumulate operations in convolution, where each pixel is repeatedly multiplied by filter weights and accumulated. Research by Sze *et al.* [1] and Redmon and Farhadi [2] indicates that multiply-accumulate operations account for over 90% of the total computational cost in convolutional neural network (CNN) based image processing, posing a significant challenge for edge devices with limited processing capabilities. Most edge devices, such as Raspberry Pi, NVIDIA Jetson, and microcontrollers, have limited on-chip memory, requiring frequent access to off-chip DRAM to store image data and convolution kernels [3]. However, external memory access is energy-intensive and introduces significant latency. Research by Horowitz [4] and Jouppi *et al.* [5] has shown that off-chip memory access can consume up to 100 times more energy than an arithmetic operation, making memory bandwidth a critical bottleneck in convolutional processing. Furthermore, power efficiency is a major constraint in edge computing, particularly for battery-powered devices. Traditional convolution operations, when executed on central processing units (CPUs), graphics processing units (GPUs), or Field-programmable gate arrays (FPGAs), demand substantial energy. Studies by Chen *et al.* [6] and Han *et al.* [7] demonstrate that deep learning workloads on embedded GPUs consume excessive power, limiting their usability in real-time edge AI applications. Many edge applications, such as autonomous driving, robotics, and video surveillance, require low-latency processing. Conventional convolution methods introduce latency due to sequential MAC computations and memory transfer overhead. Research by Gholami *et al.* [8] and Howard *et al.* [9] highlights that even optimized CNN models can exhibit significant inference latency, rendering them unsuitable for certain real-time edge AI applications. To mitigate these challenges, researchers are exploring hardware-friendly alternatives, such as binarized convolution using XNOR-Popcount operations. Instead of performing computationally expensive multiplications, XNOR-based convolution employs bitwise logic operations followed by population counting (Popcount) to accumulate results efficiently. Studies by Rastegari *et al.* [10], Courbariaux *et al.* [11], and others [12]-[15] report that XNOR-Net can achieve up to 58 times faster convolution while significantly reducing power consumption.

This study adapts the XNOR-Popcount paradigm to the Prewitt edge detector and proposes a dedicated hardware architecture for real-time edge detection. Instead of conventional arithmetic convolution, the design binarizes input pixels and applies XNOR and Popcount operations to improve efficiency. Both the XNOR–Popcount-based and a traditional MAC-based Prewitt detector are implemented in hardware and compared in terms of accuracy, resource usage, power, and speed. The contributions include: (i) a multiplier-less edge detector using 1-bit pixel and kernel representations; (ii) a functional and waveform-level comparison showing close agreement with the MAC baseline; and (iii) an FPGA implementation that significantly reduces LUTs, FFs, and DSPs while lowering dynamic power and increasing throughput. Qualitative edge maps and quantitative dissimilarity metrics indicate that, despite its simplified computation, the proposed design preserves key edge information with minor accuracy loss, supporting its suitability for practical image-processing applications.

2. PROPOSED METHOD

In conventional CNNs, edge information is extracted by sliding a small kernel (typically 3×3 or 5×5) across the image and multiplying each pixel in the receptive field by its corresponding weight, then summing the partial products to form an output feature value [16], [17]. Figure 1 illustrates this operation for common vision tasks such as handwritten-digit recognition on MNIST [18] and face recognition on ORL [19]. Although effective on power-hungry GPUs, this MAC-style convolution is burdensome on edge platforms where power budgets are often only a few milliwatts [20]. On such resource-constrained hardware, conventional convolution-based edge detection suffers from three key bottlenecks: (i) high computational load, as each output pixel requires multiple fixed-point MAC operations; (ii) substantial memory bandwidth, due to repeated access to overlapping pixel windows; and (iii) elevated energy consumption from intensive switching in multiplier trees and DSP blocks. These factors lower frame rates, raise thermal stress, and shorten battery life-particularly problematic in real-time edge-computing applications.

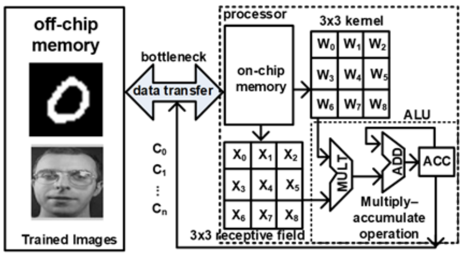


Figure 1. Data transfer bottleneck between memory and convolution core

To address these limitations, this study adopts an XNOR-Popcount-based edge detection approach, in which both input pixels and kernel weights are binarized to 1-bit values. Convolution is replaced by bitwise XNOR operations followed by a Popcount, eliminating multipliers and using only simple logic and adder trees. This greatly reduces logic utilization, critical-path delay, memory footprint, and dynamic power, while freeing DSP resources for other tasks. As a result, the XNOR-Popcount technique is well suited to low-power, resource-constrained edge devices.

2.1. Binarization process

In the first stage, each 8-bit pixel in the 3×3 window is binarized using a programmable threshold (typically 128): pixels with intensity ≥ threshold map to 1, others to 0, preserving contrast while reducing switching activity. Figure 2 contrasts this with a traditional Prewitt implementation, where each kernel sample is multiplied by its +1/–1 coefficient and accumulated to form G_x^2 and G_y^2, a MAC-heavy process in both area and energy. In the proposed method, the binarized 3×3 window is instead compared with a binary-encoded Prewitt kernel using parallel XNOR gates, and a subsequent Popcount simply tallies the ones to obtain a value proportional to the gradient—without any multipliers. Thus, convolution is replaced by lightweight bitwise logic and a small adder tree, yielding a much more efficient solution for real-time edge detection on resource-constrained devices.

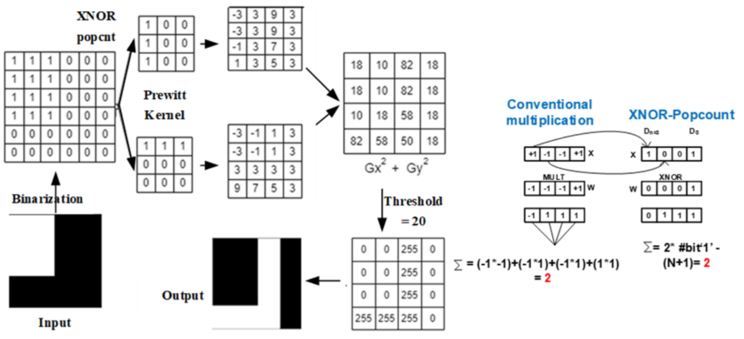


Figure 2. Conceptual overview of the XNOR-Popcount reformulation of the Prewitt edge detector

2.2. XNOR logic for binary Prewitt convolution

The binarized 3×3 window is convolved with binary horizontal and vertical Prewitt kernels (Gx, Gy), where +1 and –1 are encoded as 1 and 0. Each pixel–kernel pair goes through an XNOR gate, and logic ‘1’ indicates a contributing match, replacing signed multiplications with simple bitwise operations. For comparison, Figure 3 shows the conventional MAC core: the 72-bit bus `i_pixel_data` [71:0] feeds two multiplier arrays that compute Gx and Gy using signed Prewitt kernels. Their partial products pass through adder trees to form 11-bit sums (`sumData_Gx, sumData_Gy`), which are squared, added to obtain the gradient magnitude, and then compared to a programmable 8-bit threshold to generate `o_mac_data[7:0]` with `o_data_valid` and `o_intr`. Although fully pipelined to process one window per clock, this MAC core relies on two DSP multipliers and a larger LUT/FF footprint than the XNOR–Popcount alternative.

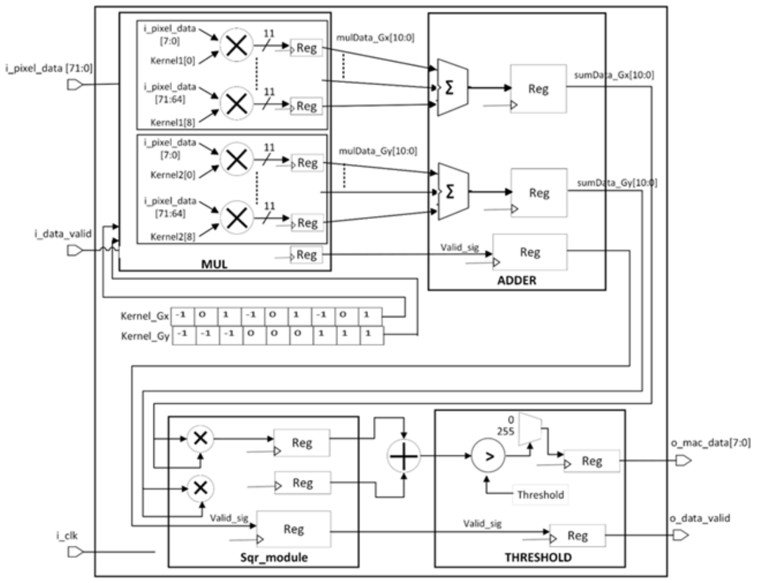


Figure 3. A conventional MAC-based core serves as the reference implementation of the edge detector

In the proposed architecture, conventional MAC-based convolution is replaced by an XNOR–Popcount scheme. First, each 8-bit grayscale pixel is binarized using a fixed threshold, preserving the contrast needed for edge detection in 1-bit form. The Prewitt kernels are likewise encoded as binary masks, mapping +1 → 1 and –1 → 0, so the original +1/–1 pattern becomes a 1/0 pattern. With both the 3×3 image window and kernel in binary form, convolution is implemented with logic gates: each pixel bit is compared to its corresponding kernel bit using an XNOR gate, which outputs 1 for matches and 0 for mismatches. For the horizontal kernel K_x, a match corresponds either to a bright pixel under a +1 weight or a dark pixel under a –1 weight, both contributing positively to the edge response, while mismatches represent negatively weighted contributions.

2.3. Popcount tree and gradient magnitude

The XNOR outputs are fed into a Popcount adder tree that counts the number of matches in each 3×3 neighborhood. The resulting counts for Gx and Gy are then squared and summed to approximate the gradient magnitude. Because the maximum value of this sum is 162, an 8-bit data path is sufficient, which further reduces logic and power. The XNOR-Popcount methodology is built on the principle of computing edge responses using only bitwise operations and lightweight counting logic, effectively eliminating the need for multipliers. To implement this approach, a dedicated hardware module, termed `XNOR_POPCNT`, was developed to perform horizontal and vertical edge detection using the XNOR-Popcount algorithm. The module accepts three input signals and produces two outputs. The `i_clk` input serves as the system clock, while the `i_pixel_data[71:0]` bus delivers a 72-bit data window corresponding to a 3×3 neighbourhood of grayscale pixels (8-bit values). The `i_pixel_data_valid` signal indicates the presence of valid input data and initiates processing. Upon completion, the resulting 8-bit edge response is output via `o_xnor_pop_data[7:0]`, with the `o_xnor_pop_data_valid` signal asserted to indicate the availability of valid output data. Figure 4 shows the internal micro-architecture of a single processing element (PE) in the XNOR_POPCNT module, implemented as a five-stage pipeline. First, in the Binarization Stage, each 8-bit pixel is compared with a programmable threshold to produce a 1-bit value, compressing the 3×3 window from 72 to 9 bits and reducing downstream switching. In the XNOR Stage, two 9-bit masks encode the horizontal (Gx) and vertical (Gy) Prewitt kernels and are applied to the binarized window via nine XNOR gates, removing the need for multipliers or DSPs. The population count stage then uses a small adder tree (depth $\lceil \log_2 9 \rceil = 4$) plus a ripple-carry adder to generate two 5-bit values for |Gx| and |Gy|. In the Square–Accumulate Stage, these gradients are squared and summed; because the maximum sum is 162, an 8-bit datapath is sufficient and helps lower dynamic power. Finally, the threshold comparison stage compares this magnitude to a configurable threshold, asserting `o_xnor_pop_data_valid` and driving the 8-bit edge output on `o_xnor_pop_data` when valid. Clock and power gating at both PE and register levels disable logic whenever `i_pixel_data_valid` is low, further improving efficiency for real-time edge detection in resource-constrained embedded vision systems.

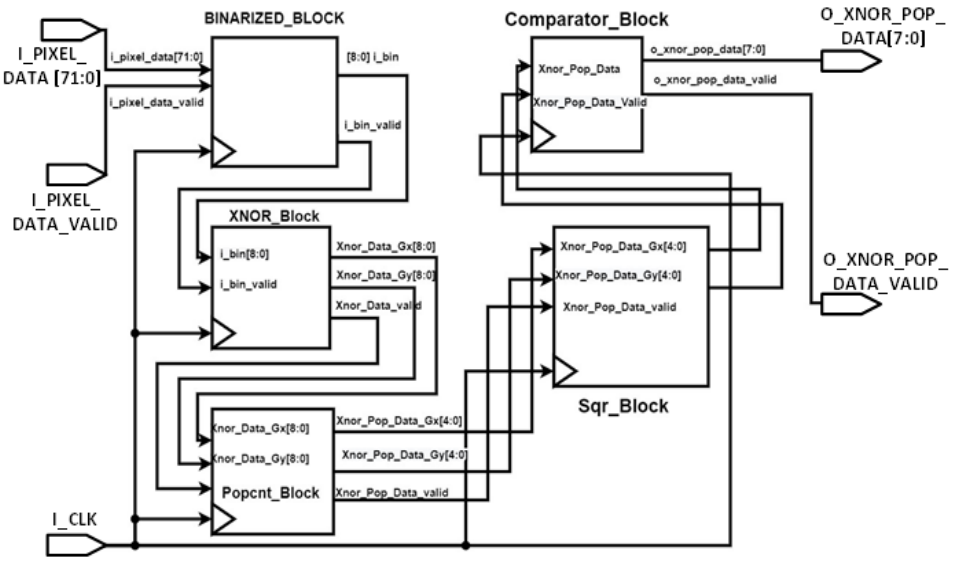


Figure 4. System-level block diagram of the proposed XNOR–Popcount (XNOR_POPCNT) core

3. ENERGY-EFFICIENT HARDWARE MODULE FOR EDGE DETECTION

The XNOR-Popcount core is integrated into a top-level architecture with line buffers, window generation, and output buffering, forming a streaming pipeline capable of processing one pixel/clock in real time. Figure 5 shows the top-level entity, Image_Processing_Top, which coordinates all datapath and control functions in the edge-detection pipeline. In this figure, 8-bit pixels `i_data[7:0]` with the handshake signal `i_data_valid` are first received by the Image_Control block, whose line buffers generate a 3×3 window on `i_pixel_data[71:0]` and assert `i_pixel_data_valid` when the window is ready. This window is then fed to the configurable XNOR–Popcount/MAC core, which can operate either as the proposed binary XNOR–Popcount engine or as a conventional MAC-based Prewitt filter. The core outputs an 8-bit edge value on `o_xnor_pop_data[7:0]` together with `o_xnor_pop_data_valid`, which are passed to the Output_Buffer FIFO. The buffer drives the external bus `o_data[7:0]`, asserts `o_data_ready` when valid data are available, and signals events such as underflow or frame boundaries via `o_intr`. A backpressure signal `i_data_ready`, derived from buffer status and core activity, ensures lossless streaming. All modules share the system clock `i_clk` and active-low reset `i_reset_n`, and use ready/valid strobes on both input and output sides for AXI-style handshaking.

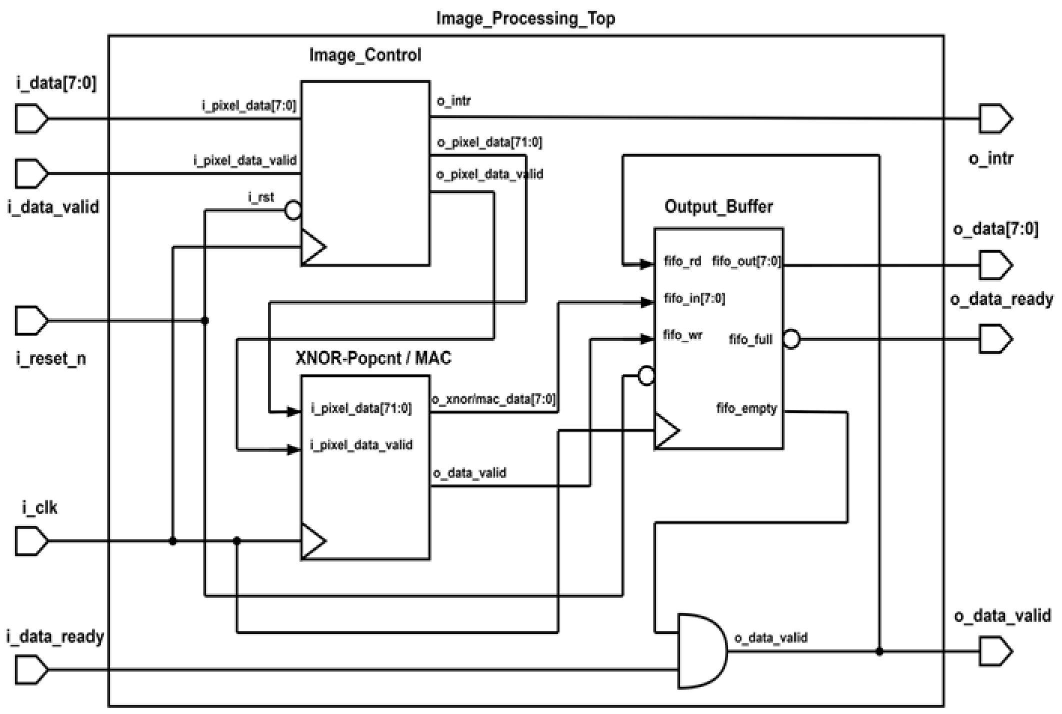
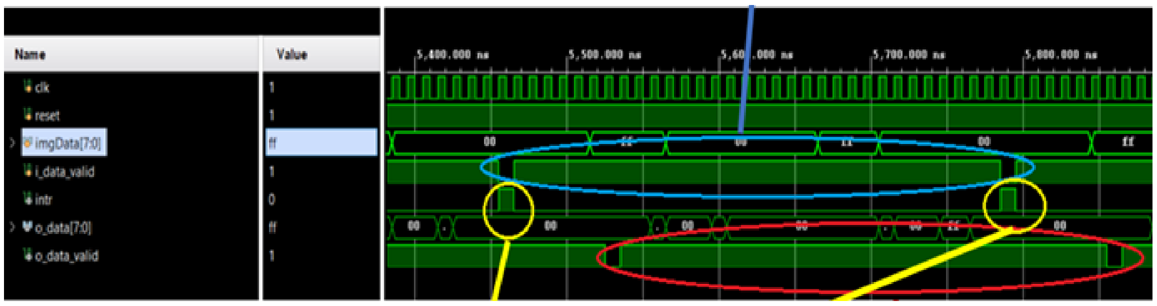


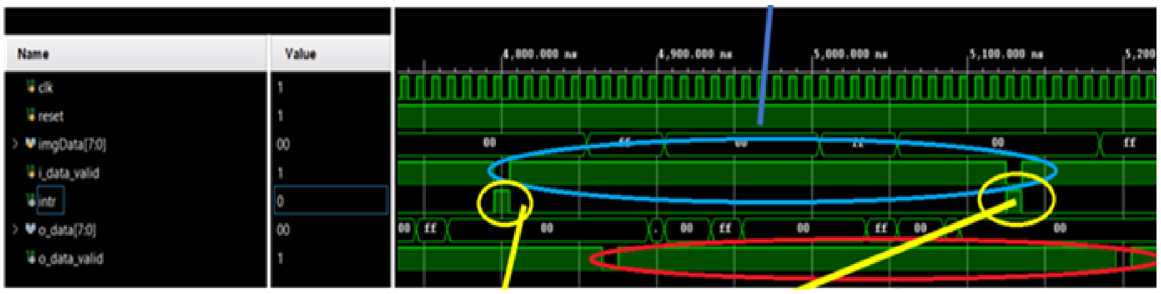
Figure 5. Top-level architecture of the proposed edge-detection module, showing the integration of the XNOR–Popcount/MAC core with image control logic and output buffering

Figure 6 compares the timing waveforms of the proposed XNOR-Popcount core and the MAC-based reference when they are driven by the same image stream. The traces show the clock, input data and its valid signal, the interrupt/handshake signal, and the binary edge outputs, allowing us to verify both functional equivalence and differences in latency and throughput. In Figure 6(a), the waveforms correspond to the XNOR-Popcount design. The highlighted regions show that the input-valid, interrupt, and output-valid signals follow a compact, regular handshake: each burst of input pixels is processed with short internal latency, and the corresponding edge outputs are produced in tightly packed bursts at 250 MHz, with no idle cycles between windows.

In Figure 6(b), the waveforms correspond to the MAC-based design under the same stimulus. Although the handshake sequence is logically identical, the deeper pipeline causes longer i_data_valid activity during line-buffer filling and visible gaps in o_data_valid as the FIFO drains. These extended and idle regions illustrate the higher latency and lower effective throughput of the MAC implementation compared with the XNOR–Popcount core.



(a)



(b)

Figure 6. Operational waveforms comparing (a) the XNOR-Popcount and (b) the MAC-based module

4. RESULTS AND DISCUSSION

In this study, the edge-detection quality of the proposed XNOR-Popcount hardware was compared against a software Prewitt operator and a MAC-based hardware implementation. Figure 7 illustrates this comparison: (a) software full-precision Prewitt, (b) MAC-based FPGA Prewitt, and (c) the XNOR-Popcount module. Visually, the XNOR-Popcount edge map is almost indistinguishable from the software result, correctly capturing all prominent edges. Minor differences are limited to very weak or noisy edges, which may appear slightly thinner or be suppressed by the binary approximation, but no major edges are lost and no obvious false edges appear. The MAC-based output matches the software result, as expected. For MNIST digits, the XNOR-Popcount core reproduces clean, mostly one-pixel-wide contours, while the MAC design tends to yield slightly thicker strokes. On ORL faces, the XNOR-Popcount core preserves facial boundaries and key features more consistently, whereas the MAC variant occasionally introduces small gaps and spurious edges, especially near hairlines.

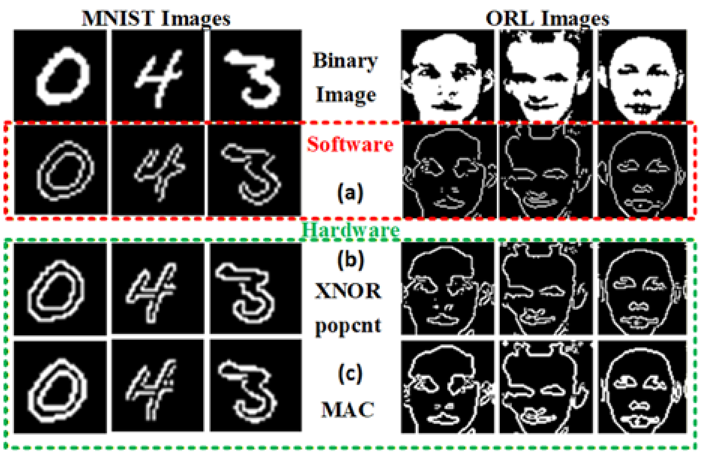
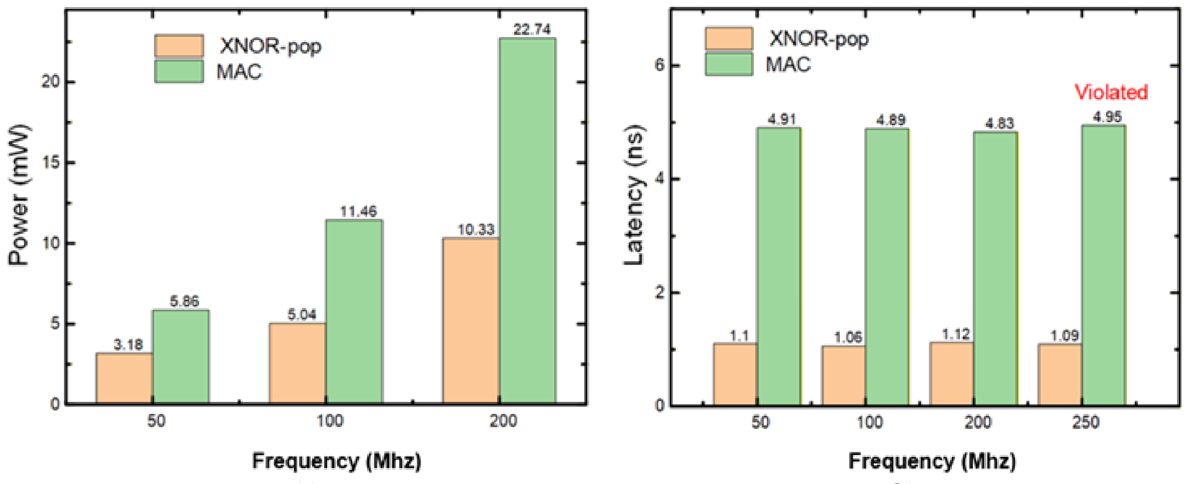


Figure 7. Edge images produced by the Prewitt filter (a) in software using full-precision convolution, (b) in FPGA hardware using MAC operations, and (c) in a hardware module based on XNOR-Popcount

Table 1 shows that the XNOR–Popcount edge detector is significantly more resource-efficient than the MAC-based Prewitt design on the target FPGA. The MAC core uses 379 LUTs, 166 flip-flops, and two DSP slices, whereas the XNOR–Popcount version needs only 172 LUTs and 123 flip-flops and uses no DSPs. This corresponds to about 55% fewer LUTs and 26% fewer FFs, plus complete removal of multiplier resources. These gains arise from replacing multi-bit MAC operations with XNOR gates, simple shifts, and two's-complement subtraction on narrow data paths. Freeing DSP slices and reducing logic not only lowers power but also shortens critical paths, enabling higher clock frequencies or additional on-chip functionality.

Table 1. Hardware resource utilization of XNOR–Popcount and MAC-based implementations

Hardware resources	LUT	Flip-Flop	DSP
MAC	379	166	2
XNOR-Pop	172	123	0



(a) (b)

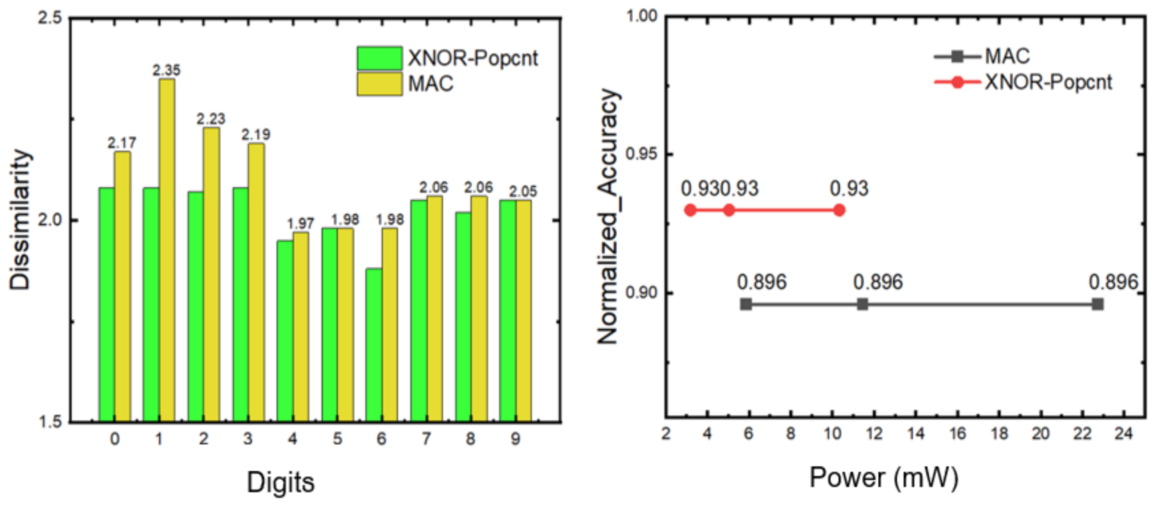
Figure 8. Implementation results for edge-detection cores (a) dynamic power versus clock frequency and (b) per-frame latency for an input image

Figure 8 summarizes the implementation results for the two architectures. In Figure 8(a), the dynamic power consumption of both designs is plotted versus clock frequency, showing that the XNOR-Popcount core consistently draws much less power-about 60% less at the evaluated operating points-because multiplier arrays are replaced by XNOR gates and a Popcount tree, greatly reducing switching activity and capacitive load. Figure 8(b) reports the per-frame processing latency for a 28×28 image; here, the XNOR–Popcount design achieves lower latency and scales more favorably with frequency thanks to its shallower datapath and shorter critical path, enabling operation at up to ~5× the maximum clock of the MAC-based core. In practice, the XNOR-Popcount engine can either process many more pixels per second or maintain the same frame rate as the MAC design while running at a lower clock.

The proposed XNOR-Popcount architecture successfully optimizes hardware efficiency at the expense of only a minor, controlled approximation in edge-detection fidelity. This design achieves substantial hardware savings, eliminating DSP multipliers and reducing LUT utilization by approximately 50% compared to the traditional MAC-based Prewitt filter. Analysis on the MNIST dataset in Figure 9(a) confirms the effectiveness of this trade-off, showing that the XNOR-Popcount edge maps maintain close fidelity, with per-digit dissimilarity tightly clustered around two. Furthermore, the XNOR-Popcount core achieves a slightly higher average normalized accuracy (~0.93) than the MAC design (~ 0.896), indicating that key edge structures are well-preserved. Quantifying the power profile in Figure 9(b), the architecture consistently maintains its superior normalized accuracy across all tested clock frequencies (50–200 MHz) while consuming only 44–56% of the dynamic power of the MAC design (e.g., 3.18 mW vs. 5.86 mW at 50 MHz). Since both designs offer the same throughput (one pixel per clock), the XNOR-Popcount implementation achieves approximately half the energy consumption per frame, enabling either greater operational efficiency or a higher achievable clock frequency under a fixed power budget.

In the context of prior FPGA-based edge-detection work, the proposed architecture occupies a distinct design point. As summarized in Table 2, it advances beyond the partial-product pruning of Perri *et al.*

[21] and the compressor retiming of Schiel and Bainbridge-Smith [22], as well as more recent Sobel/Prewitt/Roberts implementations [23]-[25] that remain within a multi-bit MAC paradigm. Different from prior works that only optimize multipliers, this study eliminates them. The XNOR-Popcount approach changes the computation model itself, accepting a controlled loss in edge fidelity in exchange for more aggressive reductions in logic and power while maintaining comparable or higher operating frequencies. Conceptually, it brings ideas from binary neural networks [10], [11], [14], [15] into classical edge detection: 1-bit pixels and kernels act as both quantization and implicit denoising, yielding edge maps suitable for embedded-vision pipelines. The resulting hardware is multiplier-less, shallow, and amenable to clock/power gating, making it a strong building block for resource-constrained vision nodes and a promising basis for extensions such as adaptive thresholds, low-bit variants, and binary reformulations of other edge operators.



(a) (b)

Figure 9. Per-digit dissimilarity and accuracy–power trade-off (a) Dissimilarity of MNIST edge maps for XNOR–Popcount vs. MAC Prewitt and (b) Average normalized edge accuracy versus dynamic power, showing higher accuracy at lower power for XNOR–Popcount

Table 2. Performance comparison with other FPGA-based edge detection designs

Studies	Schiel and Bainbridge-Smith [22]	Perri *et al.* [21]	This study
Hardware overhead	no DSP usage; 4.4% area reduction compared with prior design	22% fewer LUTs than exact Booth/Dadda equivalents	55 % LUT, 26 % FF reduction and no DSP usage compared with MAC
Power consumption	not mentioned	Up to 80 % energy saving vs exact multipliers	~60 % lower dynamic power than a MAC
Frequency (timing)	clean at 97 MHz on Spartan-3E (1.28x speed-up)	clean at 250 MHz on Artix-7 (1.8x speed-up)	clean at 250 MHz on Zynq-7000 (5x speed-up)

5. CONCLUSION

This study shows that a binarized XNOR-Popcount reformulation of the Prewitt operator can deliver high-quality edge maps at a fraction of the hardware cost of conventional convolution-based designs. By replacing 8-bit MACs with 1-bit logic, the proposed module removes all DSP usage, cuts logic utilization by about half, and substantially reduces dynamic power, while still meeting or exceeding 250 MHz for embedded vision. Experiments on handwritten digit and face datasets indicate that a small loss of fine detail is outweighed by 4-5× gains in speed and energy efficiency, making the approach attractive for resource-constrained platforms. Future work includes extending this strategy to other operators (e.g., Sobel, Laplacian), combining it with adaptive thresholding or low-bit quantization for better faint-edge recovery, and integrating the XNOR–Popcount module with lightweight CNN accelerators as a front-end feature extractor in compact embedded AI pipelines. Overall, the XNOR-Popcount paradigm offers a scalable path toward ultra-low-power, always-on vision for edge and IoT devices.

FUNDING INFORMATION

The authors state no funding is involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Van-Khoa Pham	✓	✓	✓			✓	✓		✓	✓	✓	✓	✓	
Lai Le				✓	✓			✓		✓				

C	:	**C**onceptualization	I	:	**I**nvestigation	Vi	:	**Vi**sualization
M	:	**M**ethodology	R	:	**R**esources	Su	:	**Su**pervision
So	:	**So**ftware	D	:	**D**ata Curation	P	:	**P**roject administration
Va	:	**Va**lidation	O	:	Writing - **O**riginal Draft	Fu	:	**Fu**nding acquisition
Fo	:	**Fo**rmal analysis	E	:	Writing - Review & **E**diting			

CONFLICT OF INTEREST STATEMENT

The author states no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: a tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, Dec. 2017, doi: 10.1109/JPROC.2017.2761740.

[2] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," *arXiv preprint arXiv*, Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.02767

[3] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: automatic end-to-end deployment of real-world DNNs on low-cost IoT MCUs," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1253-1268, Aug. 2021, doi: 10.1109/TC.2021.3066883.

[4] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, IEEE, Feb. 2014, pp. 10-14. doi: 10.1109/ISSCC.2014.6757323.

[5] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings - International Symposium on Computer Architecture*, New York, NY, USA: ACM, Jun. 2017, pp. 1-12. doi: 10.1145/3079856.3080246.

[6] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017, doi: 10.1109/JSSC.2016.2616357.

[7] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv*, Feb. 2015, [Online]. Available: http://arxiv.org/abs/1510.00149

[8] A. Gholami *et al.*, "SqueezeNext: hardware-aware neural network design," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, Jun. 2018, pp. 1719-1728. doi: 10.1109/CVPRW.2018.00215.

[9] A. G. Howard *et al.*, "MobileNets: efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv*, Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: imagenet classification using binary convolutional neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9908 LNCS, 2016, pp. 525-542. doi: 10.1007/978-3-319-46493-0_32.

[11] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv*, Mar. 2016, [Online]. Available: http://arxiv.org/abs/1602.02830

[12] S. Zhu, L. H. K. Duong, and W. Liu, "XOR-net: An efficient computation pipeline for binary neural network inference on edge devices," in *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, IEEE, Dec. 2020, pp. 124-131. doi: 10.1109/ICPADS51040.2020.00026.

[13] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Enabling AI at the edge with XNOR-networks," *Communications of the ACM*, vol. 63, no. 12, pp. 83-90, Nov. 2020, doi: 10.1145/3429945.

[14] S. Rasoulinezhad, S. Fox, H. Zhou, L. Wang, D. Boland, and P. H. W. Leong, "MajorityNets: BNNs utilising approximate Popcount for improved efficiency," in *Proceedings - 2019 International Conference on Field-Programmable Technology, ICFPT 2019*, IEEE, Dec. 2019, pp. 339-342. doi: 10.1109/ICFPT47387.2019.00062.

[15] F. Conti, P. D. Schiavone, and L. Benini, "XNOR Neural engine: A hardware accelerator IP for 21.6-fJ/op binary neural network inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2940-2951, Nov. 2018, doi: 10.1109/TCAD.2018.2857019.

[16] Y. Liu *et al.*, "Richer convolutional features for edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1939-1946, Aug. 2019, doi: 10.1109/TPAMI.2018.2878849.

[17] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 4905-4913, Jan. 2016, [Online]. Available: http://arxiv.org/abs/1701.04128.

[18] Y. LeCun, C. Cortes, and C. J. Burges, “The MNIST database of handwritten digits.” Accessed: Sep. 24, 2025. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[19] F. S. Samaria and A. C. Harter, “Parameterisation of a stochastic model for human face identification,” in *IEEE Workshop on Applications of Computer Vision - Proceedings*, IEEE Comput. Soc. Press, 1994, pp. 138-142. doi: 10.1109/acv.1994.341300.
[20] L. Martin Wisniewski, J. M. Bec, G. Boguszewski, and A. Gamatié, “Hardware solutions for low-power smart edge computing,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 61, Nov. 2022, doi: 10.3390/jlpea12040061.
[21] S. Perri, F. Spagnolo, F. Frustaci, and P. Corsonello, “Designing energy-efficient approximate multipliers,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 4, p. 49, Sep. 2022, doi: 10.3390/jlpea12040049.
[22] J. Schiel and D. A. Bainbridge-Smith, “Efficient edge detection on low-cost FPGAs,” *arXiv preprint arXiv*, p. 5, Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.00504
[23] N. Shylashree, M. Anil Naik, A. S. Mamatha, and V. Sridhar, “Design and implementation of image edge detection algorithm on FPGA,” *International Journal of Circuits, Systems and Signal Processing*, vol. 16, pp. 628-636, Jan. 2022, doi: 10.46300/9106.2022.16.78.
[24] N. Nausheen, A. Seal, P. Khanna, and S. Halder, “A FPGA based implementation of Sobel edge detection,” *Microprocessors and Microsystems*, vol. 56, pp. 84-91, Feb. 2018, doi: 10.1016/j.micpro.2017.10.011.
[25] H. A. T. Abdullah, R. Z. Mahmood, S. M. Alhaj Zber, R. A. Mohammed, M. R. Ahmed, and A. W. Talab, “FPGA-based three edge detection algorithms (Sobel, Prewitt and Roberts) implementation for image processing,” *Przeglad Elektrotechniczny*, vol. 2024, no. 2, pp. 29-33, Feb. 2024, doi: 10.15199/48.2024.02.05.

BIOGRAPHIES OF AUTHORS

Van-Khoa Pham received the B.S. (2010) and M.S. (2014) degrees in Computer Technology and Electronics Engineering from the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam. He earned the Ph.D. degree in Electronics Engineering from Kookmin University, Seoul, Korea, in 2019. In 2010, he joined the Integrated Circuit Design Research and Education Center (ICDREC), where he contributed to the development of the VN8-01, the first commercially designed and fabricated microcontroller in Vietnam. From May 2011 to 2021, he was with the Faculty of Electrical and Electronics Engineering at HCMUTE. He is currently a senior lecturer in the Department of Computer and Communication Engineering and serves as Head of Computer Technology Engineering in the Faculty of International Education at HCMUTE. His research interests include low-power VLSI, memory design, internet-of-things (IoT) hardware, and power IC design. Dr. Pham has published in Electronics Letters, IEEE Transactions on Nanotechnology, Journal of Semiconductor Technology and Science, Micromachines, International Journal of Computing, Indonesian Journal of Electrical Engineering and Computer Science, and at the IEEE International Symposium on Circuits and Systems (ISCAS), among others. He can be contacted at email: khoapv@hcmute.edu.vn.

Lai Le received his B.S. degree in Computer Technology from Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam, in 2023. He is currently pursuing a Master’s degree in Electronics and Telecommunication Engineering at HCMUTE. His research interests include low-power IC design. He can be contacted at email: 2430705@student.hcmute.edu.vn.